UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 10/717,747
Filing Date: November 20, 2003
Appellant(s): EICKEMEYER ET AL.

Bret J. Peterson
Reg. No. 37,417
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 18 July 2008 appealing from the Office action mailed 19 February 2008.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

No amendment after final has been filed.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claim Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

Doing (U.S. Patent No. 6,438,671) filed on 01 July 1999 and patented 20 August 2002

Parady (U.S. Patent No. 5,933,627) filed on 01 July 1996 and patented on 03 August 1999

Shoemaker (U.S. Publication No. 2003/0135711) filed on 15 January 2002 and published on 17 July 2003

Levy (U.S. Patent No. 6,314,511) filed on 02 April 1998 and patented on 06 November 2001

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 1-2 and 9-13 are rejected under 35 U.S.C. 103(a) as being unpatentable over Doing (U.S. Patent No. 6,438,671) in view of Parady (U.S. Patent No. 5,933,627).

Regarding claim 1, Doing discloses an integrated circuit processor (col 2 lines 52-53) comprising: a first instruction buffer (fig 2 reference 203) corresponding to a primary thread (col 7 lines 55-56); a second instruction buffer (fig 2 reference 204) corresponding to a backup thread (col 7 lines 60-64); a thread switch mechanism that detects when the primary thread stalls (col 14

lines 21-27), and in response thereto, swaps information stored in the first instruction buffer with information stored in the second instruction buffer (col 14 lines 4-9 and col 11 lines 51-62).

Doing fails to disclose swapping the instructions, rather than just related data.

Parady discloses an embodiment similar to Doing (col 4 lines 42-52) and a further embodiment that discloses swapping a larger quantity of state data from various shadow registers to a main register (col 5 lines 38-43 and fig. 7).

Doing would be motivated to utilize teachings of Parady to save cost by using less silicon for read ports. See Parady col 5 lines 30-37.

When a context switch of a thread occurs, there are generally two options available for saving and retrieving the state of that data. Often times, there are two (or more) groups of registers that each contain the information for a particular thread. When a context switch occurs, the processor will start executing commands using the information in register group 2 rather than register group 1. When a second thread switch occurs, the processor will similarly switch back to register group 1. This technique is shown within the instruction buffers of Doing.

A second option is also available; one group of registers can be used as a primary register group while one (or more) register groups are backup or "shadow registers". This has the advantage, as shown in Parady, of saving cost by using less silicon for read ports.

Furthermore, although the state data within the shadow and main registers do not incorporate the entire instruction per se, they typically include portions of instructions such as immediate data. Instructions themselves, of course, are no more than a certain type of state data. Context switches involving both instructions and instruction data are analogous in nature and result in similar considerations, motivations and predictable results.

Accordingly, the Supreme Court has stated "When there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill in the art has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense. In that instance the fact that a combination was obvious to try might show that it was obvious under §103." *KSR v. Teleflex*, 550 U.S. ____ (2007).

Here, we have a finite number of identified, predictable solutions within one of ordinary skill's technical grasp. It would have been obvious at the time of the invention for one of ordinary skill in the art to take the processing system of Doing and, by analogy, incorporate the teachings of Parady in such a way that the instruction buffers of Doing exchange, not just various context information, but the instructions themselves during a thread switch.

Regarding claim 2, Doing/Parady discloses the integrated circuit processor of claim 1 wherein execution of the backup thread occurs after the swap by executing at least one instruction in the first instruction buffer (col 7 line 52 to col 8 line 3).

Note that, after the instructions have been swapped, the backup instructions (considered to be "execution of the backup thread") are executed by collecting these instructions from the sequential buffer (also known as the "first instruction buffer").

Regarding claim 9, Doing/Parady discloses a method for switching between a first thread of execution and a second thread of execution in a multithreaded processor (col 7 line 52 to col 8 line 3), the method comprising the steps of: (A) providing a first instruction buffer (fig 2

reference 203) corresponding to the first thread (col 7 lines 55-56); (B) providing a second instruction buffer corresponding to the second thread (fig 2 reference 204 and col 7 lines 60-64); (C) swapping information stored in the first instruction buffer with information stored in the second instruction buffer (col 7 line 52 to col 8 line 3).

Regarding claim 10, Doing/Parady discloses the method of claim 9 wherein step (C) is performed when switching between the first thread and the second thread is required (col 7 lines 60-64).

Regarding claim 11, Doing/Parady discloses the method of claim 9 wherein step (C) is performed when the first thread stalls (col 14 lines 21-27).

Regarding claim 12, Doing/Parady discloses the method of claim 9 wherein step (C) is performed when the second thread stalls (col 14 lines 21-27).

Note that the thread mechanisms are considered to be symmetric. See claim 8.

Regarding claim 13, Doing/Parady discloses the method of claim 9 further comprising the step of executing the second thread after the swapping of information in step (C) by executing at least one instruction in the first instruction buffer (col 7 line 52 to col 8 line 3).

See claim 2.

Claims 3-6, 8, 14-19, 21 and 22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Doing/Parady in view of Shoemaker (U.S. Publication No. 2003/0135711).

Regarding claim 3, Doing/Parady discloses the integrated circuit processor of claim 1.

Doing fails to disclose a third or fourth instruction buffer that swap instructions in a similar fashion as the first and second instruction buffer.

Shoemaker discloses the technique of Simultaneous Multi-Threading (paragraph 6).

It is expected that one of ordinary skill in the art would appreciate the fact that additional parallelism in a processing system allows for additional throughput. The technique of Simultaneous Multi-Threading is well known and has the advantages of allowing "multiple threads to share and to compete for processor resources at the same time". Doing, a processing system concerned with latency during stalls would be motivated to incorporate this technique to allow "the SMT system to continue executing useful work during a cache miss".

It would have been obvious at the time of the invention for one of ordinary skill in the art to add SMT capabilities (as in Shoemaker) to the computing system of Doing/Parady in such a way that the primary and backup threads are replicated to create a "third" and "fourth" instruction buffer, also considered to be secondary primary and backup buffers.

Note that, logically, it would be obvious to simply replicate portions of the original invention of Doing/Parady to allow for SMT capabilities. For that reason, future references to third and forth buffers will be treated the same way as the first and second buffers, disclosed by Doing/Parady alone. Consequently, Doing/Shoemaker discloses the remaining limitations.

Regarding claim 4, Doing/Parady/Shoemaker discloses the integrated circuit processor of claim 3 wherein the first and second primary threads simultaneously issue instructions for execution (Shoemaker paragraph 6).

Regarding claim 5, Doing/Parady/Shoemaker discloses an integrated circuit processor (col 2 lines 52-53) comprising: a first primary instruction buffer (col 2 lines 52-53) corresponding to a first primary thread (fig 2 reference 203); a second primary instruction buffer corresponding to a second primary thread (see claim 3); wherein the first and second primary threads simultaneously issue instructions for execution (Shoemaker paragraph 6); a first backup instruction buffer (col 7 lines 55-56); a second backup instruction buffer; a thread switch mechanism that detects when one of the first and second threads stalls (col 14 lines 21-27), and in response thereto, swaps information stored in one of the first and second primary instruction buffers corresponding to the stalled thread with information stored in one of the first and second backup instruction buffers (col 7 line 52 to col 8 line 3).

Regarding claim 6, Doing/Parady/Shoemaker discloses the integrated circuit processor of claim 5 wherein the thread switch mechanism: (1) detects when the first primary thread stalls (col 14 lines 21-27), and in response thereto, swaps the first primary instruction buffer with the first backup instruction buffer (col 7 line 52 to col 8 line 3); and (2) detects when the second thread stalls (col 14 lines 21-27), and in response thereto, swaps the second primary instruction buffer with the second backup instruction buffer (col 7 line 52 to col 8 line 3).

Note that, as discussed in the combination above, the third/fourth threads (second primary and backup threads) are considered to act in a similar fashion as the first and second threads.

Regarding claim 8, Doing/Parady/Shoemaker discloses an integrated circuit processor (col 2 lines 52-53) comprising: a first primary instruction buffer corresponding to a first primary thread (fig 2 reference 203); a second primary instruction buffer corresponding to a second primary thread (col 7 lines 55-56); wherein the first and second primary threads simultaneously issue instructions for execution; a first backup instruction buffer (col 7 lines 55-56); a second backup instruction buffer; a thread switch mechanism that detects when the first thread stalls (col 14 lines 21-27), and in response thereto, begins issuing from the first backup instruction buffer, and that detects when the second thread stalls, and in response thereto, begins issuing from the second backup instruction buffer (col 14 lines 21-27).

Note that the distinction between the first and second primary threads do not matter, because the each primary thread are considered to be symmetric.

Regarding claim 14, Doing/Parady/Shoemaker discloses the method of claim 9 further comprising the steps of: (D) providing a third instruction buffer corresponding to a third thread (fig 2 reference 203); (E) providing a fourth instruction buffer corresponding to a fourth thread (fig 2 reference 204); and (F) swapping information stored in the third instruction buffer with information stored in the fourth instruction buffer (col 7 lines 60-64).

Regarding claim 15, Doing/Parady/Shoemaker discloses the method of claim 14 wherein step (F) is performed when the third thread stalls (col 14 lines 21-27).

Regarding claim 16, Doing discloses the method of claim 14 wherein step (F) is performed when the fourth thread stalls (col 14 lines 21-27).

Regarding claim 17, Doing/Parady/Shoemaker discloses the method of claim 14 wherein the first and third threads simultaneously issue instructions for execution (Shoemaker paragraph 6).

Regarding claim 18, Doing/Parady/Shoemaker discloses a method for switching between first and second threads (col 7 line 52 to col 8 line 3) of execution in a multithreaded processor (col 5 lines 10-14), the method comprising the steps of: (A) providing a first primary instruction buffer corresponding to the first thread (fig 2 reference 203 and col 7 lines 55-56); (B) providing a second primary instruction buffer corresponding to the second thread (see claim 3); (C) providing a first backup instruction buffer corresponding to a first backup thread (fig 2 reference 204 and col 7 lines 60-64); (D) providing a second backup instruction buffer corresponding to a second backup thread; (E) simultaneously issuing instructions from the first primary instruction buffer and from the second primary instruction buffer; and (F) detecting when one of the first and second primary threads stalls (col 14 lines 21-27 and col 7 lines 60-64), and in response thereto, swapping information stored in one of the first and second primary instruction buffers

corresponding to the stalled thread with information stored in one of the first and second backup instruction buffers (col 7 line 52 to col 8 line 3).

Regarding claim 19, Doing/Parady/Shoemaker discloses the method of claim 18 wherein step (E) comprises the steps of: (1) detecting when the first primary thread stalls (col 14 lines 21-27), and in response thereto, swapping information stored in the first primary instruction buffer with information stored in the first backup instruction buffer (col 7 line 52 to col 8 line 3); and (2) detecting when the second thread stalls, and in response thereto, swapping information stored in the second primary instruction buffer with information stored in the second backup instruction buffer (col 7 line 52 to col 8 line 3).

Regarding claim 21, Doing/Parady/Shoemaker discloses a method for switching between threads of execution in a multithreaded processor (col 7 line 52 to col 8 line 3), the method comprising the steps of: (A) providing a first primary instruction buffer corresponding to the first thread (fig 2 reference 203 and col 7 lines 55-56); (B) providing a second primary instruction buffer corresponding to the second thread (see claim 3); (C) providing a first backup instruction buffer corresponding to a first backup thread (fig 2 reference 204 and col 7 lines 60-64); (D) providing a second backup instruction buffer corresponding to a second backup thread; (E) simultaneously issuing instructions from the first primary instruction buffer and from the second primary instruction buffer (Shoemaker paragraph 6); and (F) detecting when the first threads stalls (col 14 lines 21-27) and swapping instructions stored in the first primary instruction buffer

with instructions stored in the first backup instruction buffer, and issuing instructions from the first primary instruction buffer (Parady col 5 lines 38-43).

Regarding claim 22, Doing/Parady/Shoemaker discloses the method of claim 21 further comprising the step of (G) detecting when the second thread stalls, and in response thereto, swapping instructions stored in the second primary instruction buffer with instructions stored in the second backup instruction buffer, and issuing from the second primary instruction buffer (Parady col 5 lines 38-43 and Shoemaker paragraph 6).

Claims 7 and 20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Doing/Parady/Shoemaker in view of Levy (U.S. Patent No. 6,314,511).

Regarding claim 7, Doing/Parady/Shoemaker discloses the integrated circuit processor of claim 5.

Doing/Shoemaker fails to disclose a pool of backup buffers.

Levy discloses the use of a pool of registers to be used for register renaming during a context switch (col 12 lines 34-45).

Doing/Parady/Shoemaker (as previously combined) has a single backup thread for each primary thread. This technique can allow for useful processing during a cache miss; however, utilizing a technique analogous to Levy would allow for "the most flexible technique for managing" the instruction buffers (Levy col 12 lines 35-36). More flexibility for thread

switching allows for better utilization of processor resources during a latency situation caused by a cache miss.

It would have been obvious at the time of the invention for one of ordinary skill in the art to allow the computing system of Doing/Parady/Shoemaker to utilize a system analogous to the register renaming system of Levy in which a pool of threads can be used to backup the primary threads, rather than just a single backup option. This way, any backup instruction buffer in the pool may be swapped with information in the first primary instruction buffer and information in any backup instruction buffer in the pool may be swapped with information in the secondary primary buffer (col 12 lines 36-41).

Regarding claim 20, Doing/Parady/Shoemaker/Levy discloses the method of claim 18 wherein the first and second backup instruction buffers are part of a pool of backup instruction buffers (Levy col 12 lines 34-45), wherein information in any backup instruction buffer in the pool may be swapped with information in the first primary instruction buffer (col 12 lines 36-41), and wherein information in any backup instruction buffer in the pool may be swapped with information in the second primary instruction buffer (col 12 lines 36-41).

Claims 7 and 20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Doing/Parady/Shoemaker in further view of Parady.

Parady also discloses a pool of backup registers (fig. 7)

For the same reasons as shown in Levy, it would have been obvious at the time of the invention for the processing system of Doing/Parady/Shoemaker (as previously combined) to utilize a backup pool of registers (as shown in Parady).

(10) Arguments

A. Background and the Primary Reference, Parody

The claimed invention deals with a multithreaded instruction system. It discusses the fetch stage of a microprocessor where instructions are gathered from buffers and sent along sequentially to later stages of the processing system to be executed.

Modern systems have the capability of running multiple threads. Basically, each thread can be thought of as a separate and largely independent program, each with its own set of sequentially executed instructions. Thread A – for example – would contain a list of instructions that can be sequentially executed to complete a particular purpose and, similarly, Thread B would contain a separate list of instructions that are to be sequentially executed for a separate purpose.

Processors do not normally execute one thread to completion before executing the second. This is because thread execution often encounters unexpected delays – delays that take significant time at which the processor would be stalled. So, instead of allowing this stall to completely halt the processor, designers have set up thread switching mechanisms that allow the processor to execute a separate thread until the delays of the original thread are resolved.

To put this in the context of our example, Thread A would be running on the processor, allowing its instructions to be sequentially fetched for execution, until an unexpected delay is

encountered. At this point, the processor would attempt to resolve this delay while simultaneously switching to Thread B and sequentially executing instructions from that thread.

As these threads are being executed, their instructions are located in sequential positions of a buffer. Thread A, for example, is located in Buffer A while Thread B is located in Buffer B. In most systems, Buffers A and B alternate as a Primary Buffer, depending on which thread is currently being executed. In particular, if the processor is currently executing Thread A, then it is fetching instructions from Buffer A (the current Primary Buffer). Then, after a thread switch, the processor begins executing Thread B by fetching instructions from Buffer B. Buffer B, then, has become the Primary Buffer and Buffer A becomes the Secondary Buffer.

This is precisely how the system in Doing works.

B. Appellant's Invention

The claimed invention of Appellant works slightly different than the general prior art discussed above. Instead of switching which buffer is the Primary Buffer, the claimed invention swaps all instructions from Buffer A and Buffer B, leaving Buffer A as the Primary Buffer but fetching instructions that are from Thread B.

C. The Secondary Reference: Doing

Appellant's invention works very similar to the general prior art above. The difference is data swapping technique. The primary question is whether this data swap is a known option that could render Appellant's invention obvious.

As it turns out, the swapping technique is known. In a typical multi-threaded system, this technique is done later down the pipeline with state data. State data is stored in a Main Register

(analogous to the Primary Buffer). This Main Register always contains the state data for whichever thread is currently running. The processing system takes the data directly from this register.

When a thread switch occurs, all the data in the Main Register is swapped with the data in one of the Shadow Registers (analogous to the Secondary Buffer). So, at any point during execution, the Shadow Registers contains all state data for the idle threads while the Main Register contains the state data for the current thread. This is how the system in Parady works.

D. The Combination

i. Examiner's Position

Appellant's invention, it appears, has done nothing more than taken the buffer execution technique commonly used for instruction fetching and incorporated the register swapping technique commonly used with state data.

The combination of Parody and Doing contains all the elements required for the rejection. The claimed invention discloses "...a first instruction buffer corresponding to a primary thread; a second instruction buffer corresponding to a backup thread..." These elements are found entirely in the primary reference Doing.

The invention further requires "...a thread switch mechanism that detects when the primary thread stalls..." Again, this element is found entirely in Doing.

The invention further requires "...in response thereto, swaps instructions stored in the first instruction buffer with instructions stored in the second instruction buffer." This is where the combination takes effect. The primary reference alone would execute from the "second instruction buffer" to gain the sequential instructions of Thread B. Once the technique of Doing

is incorporated, the buffers would swap data (similar to the swapping of state data used between the Main and Shadow Registers) in the event of a thread switch.

ii. Appellant's Argument

In Appellant's first argument, Appellant notes that the Supreme Court in *KSR* stated that there is "good reason to pursue known options." Appellant further makes the claim that the addition of the shadow registers of Parady should not be obvious because it is not a "known option". See page 8 of the Brief.

Appellant further discusses *Sakraida v. AG Pro*. From the analysis of this case, Appellant concludes that "when the shadow registers of Parady are combined with Doing in the Examiner's combination, they do not perform the same function, in the same way, to produce a similar result as taught in the prior art. The swapping of data in the shadow registers taught by Parady does not perform the same function as the corresponding structure in the claimed invention and does not yield the same results."

Appellant further states:

> *Appellant believes the Examiner has made conclusions that are not supported by the cited art. Yes, appellant believes the instruction swapping ventures into the unknown. And there is support for this statement since the Examiner has not shown it in the prior art but relies on Appellant's application to piece together the cited art. While instructions may be a type of data when stored in the computer, it does not mean that it would be obvious to modify the register data swapping hardware in the prior art to be able to handle instructions. Data and instructions may be made of the same elements but they are not the same. A hammer and a hand saw are typically made of the same elements, but they are used differently and not interchangeable. Similarly, data and instructions are not the same and not interchangeable as the Examiner's logic suggests. The Examiner's statements to the contrary, the cited art must show more than remotely similar hardware used for a different purpose to get a different result. It is the Examiner's burden to establish a prima facie case. The Appellant has shown that the Examiner's conclusory statements are lacking in factual backing and logical reasoning. Appellant respectfully requests the examiner's rejection of claims under 35 U.S.C. § 103(a) be reversed.*

See Pages 8 and 9 of the Brief. Examiner disagrees. These structures are completely analogous. The processors discussed here are multi-stage pipelined processors. This means that there are many stages of the processor that each work in parallel and each require certain data to function properly.

At the fetch stage, generally the first stage of a processor, the system requires instructions in their initial form. These instructions are gathered from a particular memory system (here, from the buffers) and later sent to further pipeline stages.

At the decode stage, which generally occurs after the fetch stage, these instructions are converted from their initial form to state data. More specifically, the instruction bits are pulled apart for their essential information – be it addresses, ALU control signals, etc. – and this information, in combination with operands and other bits, become the state data of a processor.

The remaining stages of a processor use this state data, rather than instructions within their initial form, to complete their functions. When a thread is being executed, this state data is stored in a Main Register. Once a thread switch occurs, state data for the new thread is swapped between a Shadow Register into the Main Register. Then, this state data is gathered from the Main Register by the processor and continues as usual.

Appellant's argument that the Shadow Registers of Parady are not a "known option" appears to rely heavily on the fact that state data is being swapped rather than instructions. This argument is simply without merit. As indicated above, instructions are broken down and converted into state data after the decode stage. Instructions within their initial form do not exist after that stage. So, within the context of later pipeline stages, state data *are* instructions.

Recognizing this fact, the Shadow Registers of Parady complete a function remarkably similar to the claimed invention of Appellant. They contain a Primary Storage and a Secondary Storage. The Primary Storage contains instruction information that is being currently executed by the processor while it executes Thread A. Information for Thread B is stored in the Secondary Storage. Upon a thread switch, the instruction information within the Primary and Secondary Storage locations are swapped and, subsequently, the processor executes Thread B from the Primary Storage. They work precisely in the same way. The only difference is that Appellant's claimed invention is located in the fetch stage, where instructions are found in their initial form, rather than later pipeline stages, were instructions are found only in the form of state data. See Parody col 5 lines 38-43 and fig. 7.

It follows that the state data swap, used to render the buffer instruction swap obvious, is a "known option" as discussed by the Supreme Court in *KSR*.

E. Motivation

Under *Graham v. Deere*, a rejection method is presented that uses a motivation to combine. Since that time, *KSR* noted that the *Graham* method was too rigid and that motivation may not necessarily be required.

It is helpful at this point to look at Appellant's invented and its alleged motivation for this combination.

i. Motivation for Appellant's Combination

Generally, processing systems are altered for one of the following (not necessarily exhaustive) list of reasons: speed, power, cost, and complexity.

Looking at Applicant's invention, it cannot be more efficient with speed. During a thread switch, the system of Doing simply stops executing from Buffer A and starts executing from Buffer B. This requires only for a single signal to change value to cause a multiplexer to change the execution entry from Buffer A to Buffer B.

Appellant's invention does not require this multiplexer; however, it requires an additional step of moving all the data from Buffer A and transferring it into Buffer B, and vice versa. This cannot be faster. And, indeed, Appellant does not allege speed gains.

Similarly with power, a single multiplexer switch takes up significantly less power than swapping the contents of two entire buffers. Appellant does not allege such an advantage.

With respect to cost, the requirement of hardware to swap the data between the two buffers is vastly greater than that of a single multiplexer. This hardware is costly. Indeed, no cost benefit is alleged.

Appellant, however, does make an argument of utility with respect to complexity. The following is Applicant's alleged motivation from the Specification:

> *"However, as discussed above, adding additional simultaneous threads greatly adds to the complexity of the design. In addition, the number of required registers is proportional to the number of simultaneous threads. As a result, known simultaneous multithreading techniques make handling more than two simultaneous threads very difficult and costly. Without an improved way for multithreading that supports more than two threads, the computer industry will continue to suffer from excessively expensive ways of providing more than two threads of execution in a processor."*

Examiner notes that Appellants design does not reduce the number of required registers. The same number of buffers exist (one for each thread) and the same number of registers are located in each buffer (one for each instruction within the thread). If there is an improvement in complexity, it is because there is no need for a multiplexer to control the output of more than one buffer. In Appellant's invention, the processor can look at the output of the Primary Buffer and

blindly use that output. In the Doing system, the processor needs to look at the output of both Buffer A and Buffer B and make a determination as to which output is required. This, again, is done commonly with one a select signal that is 1 bit in size – although the size would increase logarithmically as the number of buffers increases.

Appellant's design is allegedly less complex because it saves one multiplexer; however, there is an additional cost. Indeed, there needs to be circuitry to connect every register location of the Primary Buffer to every register location of each Secondary Buffer and separate multiplexers to control when the swapping occurs. This is a minimum requirement. Such a design may require even more complexity such as intermediate storage locations allowing a swap to occur within the appropriate timing.

The Patent Office within the Computer/Electrical division appears to have a low standard for the utility requirement under 35 USC 101; therefore, no such rejection has been made.

ii. Motivation for Prior Art Combination

Appellant argues that there is no motivation to combine the systems of Doing and Parady. This is an interesting argument. Since Examiner can find no benefit to Appellant's invention with respect to speed, power, cost or complexity, it follows that there is similarly no motivation for prior art to create such an invention. This is where *KSR*'s holding of a less rigid motivation rule becomes important. If the opinion of *KSR* is not intended to render wasteful inventions that have no discernable benefit unpatentable, then certainly the holding of *KSR* would be meaningless. This is precisely the situation that the Supreme Court found to be inappropriate for a rigid *Graham* test to be applied.

Indeed, when a thread switch from Thread A occurs, there are two known options available – just two ways that the instructions of Thread B can be executed; one can either start executing from Buffer B or swap both buffers and continue executing from Buffer A. Simply because Appellant has chosen the more wasteful, but equally known, option does not render Appellant's invention deserving of a patent.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejection should be sustained.

Respectfully submitted,

Brian Johnson, Patent Examiner 2183

/Brian Johnson/

/Eddie P Chan/

Supervisory Patent Examiner, Art Unit 2183

/Manorama Padmanabhan/

Quality Assurance Specialist, TC2100, WG2180